Programming 102

This is an expansion of Programming 101. Thanks to Gordon Snider for helping me organize the information in this section.

Programming 102

This presentation will attempt to answer the question "How can a non-programmer learn to write programs for OS/2?□ This presentation is not a book on programming but the Table of Contents of a book.

Programming 102

The stumbling block for new programmers is that there are so many gradients to climb right at the beginning even to get to your first executable.

And if your first executable doesn't execute where do you start looking for the problem?

Programming 102

First step -- install the OS/2 toolkit on your computer. It's on the eCS CD, and it has a wealth of information on programming OS/2.

Programming 102

Before I recommend books I will say that we've assembled a lot of books in the Warpstock giveaway. I recommend leaving some extra room because you will receive a lot of valuable items. If you can get "Real-World Programming for OS/2 2.1" or "The Art of OS/2 Warp programming", these are excellent introductions to programming in OS/2.

Programming 102

You'll want to know the C programming language. There are good books on it, but writing programs is the best way to learn C. OS/2 programming usually means C or C++ programming. The original book "The C programming language" will be available at second-hand bookstore everywhere. I also found the book "Accelerated C++" to be a good way to learn C++. The examples in the books and the toolkit are all in C.

Programming 102

Selecting a compiler from the many available. The free C compilers for OS/2, GCC and OpenWatcom, are both excellent – the best ever available for OS/2. I'm sure you will enjoy learning how to use them if you don't know already.

Programming 102

To add OS/2 functions to the basic C or C++ programs, you include OS2.H. This header file contains the entire OS/2 API. These functions are documented in the OS/2 Toolkit Information folder. But OS/2 programming books are an easier way to understand the programmers interface to OS/2.

Programming 102

To begin PM programming, it's best to start with a sample program, and dissect it. This is the approach used by the programmings books in a previous slide.

A second level would be to implement multi-threaded graphical window programming, which is something good OS/2 programs do.

Programming 102

Make is the program that automatically builds almost all OS/2 programs, including PMMail/2 and others. Make is scary to most novice programmers. Someone who is comfortable with make will be considered an expert by his peers.

More modern programs have superseded make, but OS/2 examples and programs usually are built with make.

Programming 102

Make is a program that keeps track of all your source code and builds it into the finished product. It calls the compiler and linker to do this. The advantage of make is that you can change one or more source code files, and make will do what is needed to build the program. The details of running the compiler an linker are buried in the makefile.

Programming 102

To obtain a complete listing of all the compiler switches, invoke the compiler without any argument and pipe the result to a file. That file will have a good list of the compiler switches. The help file for OpenWatcom has complete help in the User's guide. However, many of the compiler switches are quite arcane and usually not relevant to a beginning programmer.

Programming 102

All programs should have some form of help. Help files and books are created by the Information Presentation Facility. The IPF Compiler is included in the toolkit.

Programming 102

Then there is the knowledge of OO programming. That's a whole other branch. Your books will help, and the toolkit, too. You will learn about classes and objects, and how to make your own classes based on classes already present in OS/2.

Programming 102

If your first executable doesn't execute where do you start looking for the problem? I've left debugging for last. Makefiles usually have a way to build the program for debug by setting compiler and linker switches. At that point, you run your program inside the debugger. There is a skill to using the debugger, but I've found that the easy way to use a debugger is to run your program until it crashes and then examine the wreckage with the debugger.

Programming 102

More difficult to debug is the program that does nothing, but doesn't crash either. There the art of using the debugger to set break points and watch points which will reveal what is going on at any given point in the code.

Programming 102

Finally, the art of writing exception handlers and interpreting process dumps will enable you to find bugs that only occur on your user's computers.

Programming 102

How to install each of these and 'gotchas' to watch for. How to check install is correct.

Should also be a plan for mentoring by email after Warpstock is over.

Programming 102

Subversion, CVS and other source code tools. These aren't important when learning to program, but they are vital to participating in a multi-programmer project. We have terrific versions of these, but I'd like to see a session on mercurial, which I have not yet used.

Gordon pointed out that he doesn't agree about Subversion not being important when learning to program. He has already had REXX projects fail because he couldn't keep track of the different versions I had created.

Programming 102

Subversion has two parts, a server that provides access to the repository of stored source code, and the client that allows access to the server. We have recent ports of the server, and both command line client access and a marvelous client called SmartSVN.

Programming 102

Subversion keeps source code in a repository. Every time you store a changed version of a source code file in the repository, Subversion keeps track of the changes. You can retrieve any version of the source file, even if you've changed it hundreds of times. If you have many source code files, you can go back and pull out the state of your project from any point in history. This can be useful when the current version of software doesn't work, but an old version does.

Programming 102

Programmer's editors are important. There seems to be one editor per programmer. I use Visual SlickEdit, which was abandoned with an informal policy that OS/2 users may freely use the last version that was available. There are too many to list in this e-mail. But all they do is edit plain text files. Visual SlickEdit also tags files, and I can right click on anything in the source code and go to the definition of that thing. We could do a session just surveying programming editors.

Programming 102

OS/2 programming books are old. When really old books have examples in the C language, it's possible that some of them have old-style function declarations.

This is an old-style declaration from Kernighan and Ritchie, 1978.

```
main(argc,  argv)
int  argc;
char  *  argv[];
```

Now here is the same example from the second edition of the same book, 1988.

```
main(int argc, char *argv[])
```

Some compilers will take the old-style declarations if the proper compilation options are used.

Programming 102

Thanks, now go write a program.